Letter to the Editor

Smedinga, R.

Published in:
Discrete event dynamic systems-Theory and applications

Publication date:
1993

Discrete Event Dynamic Systems: Theory and Applications 3, (1993): 317–321

Letter to the Editor

R. SMEDINGA
Department of Computing Science, University of Groningen, The Netherlands email: rein@cs.rug.nl.

I had forgotton how incredibly and unforgivably sloppy the average mathematician is, how inconsistent about his syntax and how vague about the scopes of definitions and quantifications. I love mathematics, but it's mathematicians I cannot stand, for since ALGOL60 there is no longer an excuse.

E.W. Dijkstra (Lecture notes EWD1151, 28 January 1993)

In volume 2, numbers 3/4 of this journal a paper of mine was published. Because the final layout of the proofs was different from what I used in the original manuscript (due to miscommunication between the editor and myself), and did not satisfy me, I am writing this letter to explain, in general terms, why I think some notation should be preferred above others.

1. Sets

In the early days, sets were denoted by giving all its elements, such as $\{1, 2, 3, 4, 5, 6\}$. When this became impossible (too many elements or variable numbers), another notation was (and still is) used: $\{x \in \mathbb{N} \mid 1 \leq x \leq 6\}$. The general form of this notation is $\{x \in X \mid P(x)\}$, where $P(x)$ is some condition on the elements from X that should be in this set. The symbol x in this notation is a dummy that can be changed easily as long as the replaced symbol did not occur before in $P(x)$. x also denotes all the elements of the set. This might lead to misunderstanding. For example, the set of all squares of natural numbers, with these natural numbers being greater than 2, can be denoted by $\{i^2 \mid i > 2\}$. Similarly we could write $\{i^3 \mid i > 3\}$, which leads us to considering sets of the form $\{i^j \mid i > j\}$. Now we have been nicely led down the garden path! If we would have written dummies and elements separated. e.g., $\{i : i > 2 : i^2\}$ for the first and $\{i : i > 3 : i^3\}$ for the second, we would have ended up with $\{i : i > j : i^j\}$ which is completely different from the set $\{i, j : i > j : i^j\}$. In the first set j is no dummy, but a *globally defined* value, while in the second set j is also a dummy, giving a set that has elements i^j for all i and j satisfying $i > j$. So the better way is to follow the rules of making a computer program: first declare which variables are to be used (introducing the dummies), next give the type of each variable and the possible restrictions (conditions on the dummies), and, last, give the result.

2. Quantifiers

If we want to express conditions on all elements of a set using $\forall$ or express that there is an element in the set for which some condition holds, using $\exists$, the same idea as with sets can be used:

$$(\forall i : R(i) : P(i)) \qquad (\exists j, k : R(j, k) : P(j, k)).$$

Again, first dummies are introduced, next the range of those dummies is given (in R), and finally the expression which should be quantified is given (in P).

To say that all elements in a set X are positive can now be done as follows:

$$(\forall x : x \in X : x > 0).$$

The same notation can be used for other quantifications: the expression

$$(\Sigma\ i, j : i > j : f(i, j))$$

stands for

$$\sum_{i>j} f(i, j),$$

but in this last form it is not clear which are the dummies. Also, the last summation ties the notion of summation to a range of consecutive integers. The reader is invited to use this notation to express the sum of all prime numbers that are less than N.

Too often, a variant of the second form of sets is used here, e.g.,

$$(\forall i \in I)(\exists j \in J) f(i, j) \wedge g(i).$$

This last expression, however, is ambiguous and can be read as

$$\big(\forall i : i \in I : \big(\exists j : j \in J : f(i, j) \wedge g(i)\big)\big)$$

or

$$\big(\forall i : i \in I : \big(\exists j : j \in J : f(i, j)\big) \wedge g(i)\big),$$

which is quite different from the first possibility if $g(i)$ contains a globally defined j.

3. Fuzzy Texts

Expressing mathematical properties using a natural language easily leads to ambiguity. For example, words like "not," "and," "or" lead to fuzziness: "not" lacks finding a scope:

"not retired or disabled," "not uniformly convergent"; "or" is commonly used as exclusive or: "should I marry Jane or Ann?" (marry Jane and Ann is not allowed, although in "$x < 10$ or $x > 5$" both parts may be true); "and" can be used as conjunctive: "retired and disabled," but also as enumerative: "men and women." The only conclusion can be to use the natural language only for informal talk about a problem, derivation, and solution, but use formal deduction if final ends should meet.

4. Proof Layout

The last, but most important part of this letter, is devoted to the layout or proofs of lemmas, theorems, etc. In mathematics the most commonly used form of proofs is the mixture of formulas and a lot of English (or other natural language) text. As stated above, this results in ambiguous proofs because a natural language is not formal enough and easily leads to misinterpretations, and it is not at all clear if the proof is sound. If you are not sure of every step in your proof, write it down in this mixture of English and formulas and no reader will ever complain!

More and more, however, another kind of proof can be found in papers, namely the one in which logical deduction and formula manipulation are the important aspects. Such proofs merely look like

$$\begin{aligned} A &= B \\ &\Rightarrow C \\ &= D, \end{aligned}$$

to prove that $A \Rightarrow D$, instead of first showing that $A = B$, then showing that $B \Rightarrow C$, and at last showing that $C = D$ (where expressions B and C are to be written twice!). Sometimes steps are quite complex and need some hint to explain why it can be made:

$$\begin{aligned} A &= B && [\text{hint why } A = B] \\ &\Rightarrow C && [\text{hint why } B \Rightarrow C] \\ &= D && [\text{hint why } C = D] \end{aligned}$$

The following two proofs are equivalent:

$$\begin{aligned} A &= B && [\text{hint 1}] \qquad & D &= C && [\text{hint 3}] \\ &\Rightarrow C && [\text{hint 2}] & &\Leftarrow B && [\text{hint 2}] \\ &= D && [\text{hint 3}]; & &= A && [\text{hint 1}]. \end{aligned}$$

However, they do not look alike. The placement of the hints disturbs the symmetry here. If we change the layout to

$$\begin{array}{ll} & A \\ = & \quad \text{[hint 1]} \\ & B \\ \Rightarrow & \quad \text{[hint 2]} \\ & C \\ = & \quad \text{[hint 3]} \\ & D, \end{array}$$

it is absolutely clear to which operator the hint points and, moreover, this proof can also easily be read from bottom to top, which helps both reader and writer in finding the best way to prove $A \Rightarrow D$: start with A or start with D.

By putting operators in the first column, operands in the second, and the hints indented in the second column, it does not matter if some of the operators in the first column also appear in one of the operands. Compare the following two proofs (with the assumption $y = 2x$):

$$\begin{array}{ll} & x = y + 2 \\ = & \quad [y = 2x] \\ & x = 2x + 2 \\ = & \\ & x = -2, \end{array}$$

$$\begin{array}{ll} x = y + 2 & = x = 2x + 2 \qquad [y = 2x] \\ & = x = -2. \end{array}$$

In the second derivation the first line is logically equivalent to

$$x = (y + 2 = x) = 2x + 2,$$

which is nonsense because $y + 2 = x$ results in a Boolean value and $2x + 2$ does not. Also, writing

$$\begin{array}{ll} & x = y + 2 \\ = & x = 2x + 2 \qquad [y = 2x] \\ = & x = -2 \end{array}$$

does not make clear where the hint $y = 2x$ is needed.

Proving things using logical deduction and formula manipulation has another advantage. By deducing D from A, the meanings of the expressions in the steps in between need not be known. Some expression E halfway through the deduction does not necessarily have to have a useful meaning in the theory. It is just a derived formula. As long as the first and last expressions have a meaning in the theory, we can, in the proof, temporarily forget about the meaning of the formulas and concentrate on pure symbol manipulation, which enriches the possibilities in proofs and may lead to results sooner.

Acknowledgment

First of all I want to thank the editors for giving me the opportunity to make some remarks about notation. Much more about notation can be found in van Gasteren [1988] and of course in work of the man who initiated the introduced notation, Prof. E.W. Dijkstra (e.g., see Dijkstra [1982]). (It is strange to notice that pure mathematical notation had to be initiated by a computer scientist.) Some of the examples in this letter are from Dijkstra [1986]. In, for example, Backhouse [1986], a nice introduction to this notation can be found.

References

Backhouse, Roland C. *Program Construction and Verification*. Englewood Cliffs, NJ: Prentice Hall.
Dijkstra, E.W. Predicate transformers, Lecture notes EWD835, Eindhoven University of Technology.
Dijkstra, E.W. On notation, Lecture notes EWD950a, Eindhoven University of Technology.
van Gasteren, A.J.M. On the shape of mathematical arguments, Ph.D. thesis, University of Eindhoven.